CureMD

Explainable AI (XAI) Interface for MedicalMind

by Syed Hamail Hussain Zaidi

Capstone Project Report

AIMed

SUPERVISOR

Dr. Muddassar Farooq

March 28, 2023

Table of Contents

List of Figures

List of Tables

1 Abstract

The black box nature of traditional AI models has raised concerns about accountability, bias and trust in the technology. This project aims at developing an interpretability interface for a machine learning model that predicts cancer treatment failure probability. Model agnostic approaches of Explainable AI are used to achieve this, including Shap and counterfactuals explanation techniques. The data sets of different cancer types are analyzed and cleaned to remove outliers and dummy data. New models are trained on imputed data and performance is observed. Local and global explanations are generated using Shap, counterfactuals, and permutation feature importance methods. Based on the results, the least important features are dropped, and the models are retrained and compared to the original models. The final product is a front-end app that features three main components: models comparison, global and local explanation interfaces. Overall, the project aims to create an explainability layer between model and the user, thereby enabling better decision-making and understanding of the underlying factors that contribute to cancer treatment failure.

2 Background

2.1 Artificial Intelligence (AI)

Artificial Intelligence (AI) refers to the ability of machines or computer systems to perform tasks that typically require human intelligence, such as learning, reasoning, and problem-solving. AI technology utilizes algorithms and statistical models to analyze and interpret complex data, allowing machines to learn and make decisions based on that information. AI has the potential to transform virtually every industry and aspect of our lives. It can automate repetitive and mundane tasks, freeing up time for humans to focus on more creative and strategic work.

2.2 Applications of AI

Artificial Intelligence (AI) has numerous applications across various industries. Here are some examples of AI applications [1]:

- Natural Language Processing (NLP): AI is used to process and analyze human language, enabling applications like chatbots and voice assistants.
- mage and Video Recognition: AI can recognize and analyze images and videos, allowing for applications like facial recognition and object detection.
- Autonomous Vehicles: AI is used in self-driving cars to analyze data from sensors and make decisions about driving behavior.
- Healthcare: AI is used in medical imaging and diagnostics, as well as drug discovery and personalized medicine.
- E-commerce: AI is used to personalize recommendations for customers based on their browsing and purchase history.

These are just a few examples of many applications of AI as this technology continues to advance.

2.3 AI in Health Care

Artificial intelligence (AI) is revolutionizing healthcare systems by improving patient outcomes, reducing costs, and increasing access to care. AI is being used in various applications in healthcare, including medical diagnosis, drug discovery, and personalized treatment planning.

One of the most significant benefit of AI in healthcare is its ability to improve accuracy and precision. Machine learning algorithms can learn from large datasets, identifying patterns and making predictions with a high degree of accuracy. AI-powered diagnostics can detect diseases earlier by analyzing large amounts of medical data, including medical images, to identify patterns and anomalies that may be missed by human doctors. This allows for earlier detection of diseases and quick response to it.

3 Introduction

3.1 Motivation

The ‘black box’ nature of traditional AI models has raised concerns about accountability, bias and trust in the technology. The explainability of machine learning models is of great importance in healthcare systems for various reasons. Machine learning models used in HealthCare systems directly impact patient health. If these models make incorrect or biased predictions, patient safety can be compromised.

So, the stakes are high in healthcare, and the potential consequences of inaccurate predictions or decisions can be life-threatening. Therefore, it is crucial to ensure that healthcare professionals can understand how a model arrived at a decision or prediction. This enables them to assess its accuracy and trustworthiness, ensuring patient safety.

Ethical considerations are also paramount in healthcare, and patients have the right to know how decisions affecting their health are being made. An explainable model allows healthcare professionals to explain and justify the reasoning behind the model’s decisions, making them more transparent and trustworthy.

3.2 Objective

The project aims to enhance the cancer treatment failure prediction model by adding an interpretability layer. The main objective is to bridge the gap between the model and the doctor. The interpretability layer will facilitate doctor’s understanding of the prediction process. Trust is crucial for doctors to make informed decisions about patient treatment. The explainability interface will enable doctors to verify the prediction and identify any potential errors. In this way, doctors will have greater confidence in using the model to assist in patient care. The ultimate goal is to increase the doctor’s trust in the prediction model.

3.3 Challenges

Some of the challenges in the exlainability of TF prediction model are discussed below.

1. Selection of appropriate model agnostic approaches that can provide meaningful explanations to doctors.
2. Generation of explanations that are easy for doctors to understand, as they may not be familiar with the internal workings of the model or explainability. The use of complex or technical explanations may not be suitable in this context.
3. Also, It is crucial to find ways to enhance doctors’ experiences while generating explanations in maximum detail.
4. What types of graphs would be suitable for doctors? As they may have different preferences and requirements when it comes to visualizations.

3.4 Problem Statement

For this project, following goals have been set:

- Provide insights of the datasets of all cancer types
- Develop an interpretation module for explaining treatment failure (TF) predictions made by Boosted Random Forest model
- Use model agnostic approaches i.e. SHAP, Counterfactuals etc. to explain predictions made by the model.
- Generate both local and global explanations.
- Build a tentative interface using Flask/Django web framework.
- Provide feedback on the model being used for the TF prediction using explainablity.

4 Literature Review

4.1 Scope of Explainability

Algorithm of a model makes the predictions and every step of this process can be evaluated in terms of explainability and transparency. [3]

1. **Algorithm Transparency:** The understanding of what relationships a machine learning model has learned and how the model generally makes predictions is known as algorithm transparency. It is independent of the data or the learned model in the end, rather it is general understanding of the algorithm of machine learning model that we are using.
2. **Global Interpretability:** The explanation of how the learned model makes any prediction is global interpretability. It depends on the algorithm, data and the model itself. This includes, what kind of features are important to the model, what interactions take place between them. In practice, complete understanding of global interpretability is not possible because of the number of features involved in the model. Imagining feature space of more than 3 dimensions is incomprehensible for humans.
3. **Local Interpretability:** It is also important to know why the model made a particular prediction for an instance. The behavior of model might be different as a whole and for an instance. Locally, few features can have a high influence on a prediction, hence more important. For example, price of a house may have a non linear dependeance on its size but for an instance it can have linear relation with the size feature. Local interpretations are more accuracte than the global explanations.

4.2 Types of Explainablity

Different complex AI systems may require a different type of explanation. Research shows that there are six most common explanation types in XAI systems. [7] These types are discussed below.

1. **How Explanations** give a general overview of how the model is making predictions. Model graphs and decision boundaries commonly generate such type of explanations. It is also possible for user to create a mental model of the algorithm using *How* explanations.
2. **Why Explanations** explain why a prediction was made for any instance. Feature attributions convey *Why* explanations. These attributions can be generated by model-specific or model agnostic approaches.
3. **Why-Not Explanations** communicate why a particular class was not predicted for an instance. this kind of explanations are also called *Contrastive Explanations* which tells why there is a difference between the model's output and user's expectation. Feature attribution is normally used to generate such *Why* and *Why-not* explanations.
4. **What-if Explanations** give an insight to how changing model parameters, algoirthm, input instances will affect the model's output. Parameter tuning and examining the model's behavior can give *What-if* explanations.
5. **How-to Explanations** aim to communicate what hypothetical adjustment in the input instance may have resulted in a different outcome. Ad-hoc and model agnostic approaches can generate *How-to* explanations.
6. **What-Else Explanations** involve showing users examples from data set that produce similar results or outputs from the model. Such explanation type is also called *Explanation by Example.*

4.3 Approaches to Explainability

Different approaches can be opted to achieve explainability of a machine learning model. One way is to use only those algorithms that generate interpretable models like linear regression, logistic regression and simple decision trees. One limitation to use interpretable models is their inability to learn complex decision boundaries. To cater this problem, there is a need to develop explainability approaches for complex models. Such techniques are knows as Model-agnostic methods of interpretability. These approaches are independent of the model being used. Separating the model from the explanations has some advantages. It develops a flexible XAI interface where developers can use any machine learning model.

There are two types of model agnostic explanations: Global explanations, and local explanations. Global Explanations are given by Permutation feature importances, Partial dependence plots (PDPs), Global surrogate, and feature interaction. etc while Local explanations can be generated using Shapely values, Local surrogate (Lime), Counterfactuals etc. [3]

4.4 Shap

SHAP (SHapley Additive exPlanations) is a model-agnostic approach to interpreting machine learning models. It provides a unified framework to explain the output of any model by attributing the prediction to the input features, and quantifying their contribution to the prediction. The basic idea behind SHAP is to use game theory and the concept of Shapley values to assign values to each input feature, representing its contribution to the final prediction. The Shapley value is a way of distributing the total gain or cost of a group effort among the participants based on their individual contributions. In the context of machine learning, the "players" are the input features, and the "gain" is the output of the model. Formally, the Shapley value for feature j can be defined as follows:

$$\phi_j = \frac{1}{M} \sum_{S \subseteq \mathcal{F} \setminus j} \frac{|S|!(|\mathcal{F}| - |S| - 1)!}{|\mathcal{F}|!} [f(S \cup j) - f(S)]$$

where $\mathcal{F}$ is the set of input features, M is the total number of possible subsets of $\mathcal{F}$, f is the model function that maps a subset of features to a prediction, and $|S|$ denotes the number of features in subset S. The term $[f(S \cup j) - f(S)]$ represents the change in the model's output when feature j is added to the subset S. Intuitively, the Shapley value for feature j can be interpreted as the average marginal contribution of j across all possible coalitions of features. It measures how much the prediction changes on average when feature j is included, taking into account all possible combinations of other features.

Variants of Shap: There are different variants of SHAP for different types of models. Each of these implementations is designed to handle different types of models and data, and they have different strengths and weaknesses. [5]

1. **TreeShap:** is designed for computing SHAP values for models that are based on decision trees, such as Random Forests, Gradient Boosted Trees, and XGBoost. TreeShap works by recursively partitioning the feature space using the decision tree structure and computing the SHAP values for each partition. TreeShap is implemented in the shap.TreeExplainer module in the SHAP library.
2. **KernelShap:** is designed for computing SHAP values for models that are based on kernel methods, such as Support Vector Machines (SVMs) and Gaussian Processes. KernelShap works by generating samples from the distribution of feature values and computing the SHAP values for each sample using a local linear approximation. This method is accurate and flexible for non-linear models and non-Gaussian data, but it can be computationally expensive for large datasets and high-dimensional feature spaces. KernelShap is implemented in the shap.KernelExplainer module in the SHAP library.
3. **DeepShap:** is designed for computing SHAP values for models that are based on deep neural networks, such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs). DeepShap works by approximating the model with a simplified model that is based on the feature inputs, and computing the SHAP values for the simplified model. This method is scalable and accurate for large and complex models, but it can be computationally expensive for very large datasets and deep architectures. DeepShap is implemented in the shap.DeepExplainer module in the SHAP library.

Shap Plots: Here are some different types of plots that can be generated using SHAP:

1. **Feature Importance Plot:** This plot shows the relative importance of each feature in the model's output, based on the SHAP values assigned to each feature.
2. **Summary Plot:** This plot provides a summary of the SHAP values for each feature across all instances in the dataset, allowing for an overview of the relationship between each feature and the model's output.
3. **Dependence Plot:** This plot shows how the value of a single feature affects the model's output, while taking into account the values of other features. It can reveal non-linear and interactive relationships between features and the model's output.
4. **Interaction Plot:** This plot shows the interaction between two features and their impact on the model's output, using contour lines to indicate the regions of the plot where the output is high or low.
5. **Force Plot:** This plot shows the contribution of each feature to the model's output for a single instance, allowing for a detailed explanation of the model's decision-making process.
6. **Waterfall Plot:** This plot breaks down the model's output for a single instance, showing the contribution of each feature to the final prediction in a sequential manner.

4.5 CounterFactuals

Counterfactuals are a technique used in interpretability of machine learning models that aims to explain why a particular prediction was made by providing an alternative input that would change the prediction. A counterfactual explanation can help a user understand what changes they would need to make to the input in order to get a different output.

The idea behind counterfactuals is to find a minimal set of changes to the input that would change the prediction from the original class to the desired class. Formally, let f be the complex model to be explained, and let x be the input instance that produced a prediction $y = f(x)$. A counterfactual explanation aims to find an input x' that is as close as possible to x, but produces a different prediction $y' = f(x')$. The minimal set of changes can be found by minimizing a distance metric between x and x' subject to the condition that $f(x') \neq y$ and that the changes are minimal.

Dice-ml module that implements this technique accepts a couple of design parameters that can be set by the user. [6] Different parameters that can be set and tuned are discussed below.

- **query_instance:** This parameter specifies the instance for which the counterfactuals are to be generated.
- **total_CFs:** This parameter specifies the total number of counterfactuals to be generated.

- **desired_class:** This parameter specifies the desired output class for the counterfactuals.
- **penalty_coefficient:** This parameter controls the trade-off between the similarity of the counterfactuals to the query instance and their proximity to the decision boundary.
- **max_iter:** This parameter specifies the maximum number of iterations to be used for generating the counterfactuals.
- **verbose:** This parameter controls the level of verbosity of the method's output.
- **permitted_range:** It allows user to input practical ranges of each feature to generate feasible counterfactuals.
- **features_to_vary:** This parameter allows to pick the features that user wants to perturbate to generate a counterfactual.

4.6 Permutation Feature Importance

Global Permutation feature importance is a method used to compute the importance of features in a machine learning model. It works by randomly shuffling the values of a feature and observing the effect on the model's performance. The drop in performance after the shuffle indicates the importance of the feature to the model. The greater the drop in performance, the more important the feature is to the model. [4] Mathematically, the Global Permutation feature importance can be computed as follows: Let X be the feature matrix of shape (n-samples, n-features), y be the target variable of shape (n-samples,), and f(X) be the model's prediction function.
Compute the baseline score of the model on the test set:

$$score_0 = f(X)$$

For each feature i, shuffle its values randomly among all samples to get a new feature matrix X'. Compute the new score of the model on the shuffled test set:

$$score_i = f(X')$$

Compute the importance of feature i as the drop in score compared to the baseline score:

$$importance_i = score_0 - score_i$$

Normalize the importance scores by dividing by their sum:

$$importance_i^{norm} = \frac{importance_i}{\sum_{j=1}^{n_features} importance_j}$$

The resulting normalized importance scores represent the relative importance of each feature in the model. A higher score indicates a more important feature, and a lower score indicates a less important feature. Global Permutation feature importance is a simple and model-agnostic method for computing feature importances, but it can be computationally expensive for large datasets and models with many features.

5 Methodology

The methodology adopted was to analyze datasets of each cancer type thoroughly and uncover hidden trends, anomalies or other insights. Based on the insights, shap and counterfactuals were implemented to generate local explanations and Permutation feature importances for the global explanations.

5.1 Data Preprocessing

The raw datasets were in the form of python lists and it was hard to navigate and remember what is placed against what index. Source files needed to be restructured into python dictionaries and datasets into different data type i.e. pandas dataframes for better handling and manipulation. Different splits of datasets including training data, validation data, and test data were converted into dataframes using feature names as columns. Final format is a dictionary containing five keys i.e. 'training-set', 'val-set', 'test-set', 'model' and 'feature-set'.

5.2 Dataset Analysis

The datasets of each cancer type are analyzed thoroughly to detect any abnormal and hidden trends or any bias in the data. Dataset insights are very important to make sure that model has learned the right trends and is not biased to any specific feature of the dataset. It is also essential to produce explanations that make sense. In the analysis, distribution of the each feature of each cancer type is observed to check if there are any outliers or skewed data. To remove the outliers, python's multivariate imputer model is used for the imputation instead of directly imputing mean of the feature values which optimistically assumes that features are independent of each other.

5.3 Local Explainability

To develop local explainability interface, two model agnostic approaches are used i.e. Shap and CounterFactuals. The details of both are given below.

Shap: To implement Shap explanation, Shap python module is used. Different explainer algorithms such as TreeShap, KernelShap, DeepShap etc. are analyzed and KernelShap is then used to generate Shapely Values. DeepShap algorithm is optimized for deep learning algorithms and TreeShap is specifically for tree based models. Though, we are using boosted Random Forest model for TF which is a tree based model but TreeShap does not have an implementation for Random Forest. Three different types of explanations are then generated from Shapely values.

- Feature Importances in percentages
- Features responsible for increase in plan failure probability
- Features responsible for decrease in plan failure probability

CounterFactual: To generate counterfactuals, DiCE (Diverse CounterFactual Explanation) python module has been implemented. The module also has different algorithms to find counterfactuals. *random* sampling algorithm is used as it is a simple and efficient method for finding counterfactuals. In this method, the algorithm randomly samples inputs from the search space and checks if they satisfy the constraints. The search space is defined as the space of inputs that are close to the original input, subject to the constraint that they produce a different prediction.

For perturbation of feature values, it is important to distinguish categorical and continuous features. To avoid hard-coding in the code, continuous features are being picked intuitively. If the feature has more than 10 unique values, it is assumed to be continuous. 4 different counterfactuals are being generated keeping in mind the processing time. Only planning features are then picked from the generated counterfactuals and new plan failure probability is computed using the TF model and decrease in the failure probability is recorded.

5.4 Global Explainability

Permutation Feature Importance and Global Shapely Values are used to explain the model as a whole.

Permutation Feature Importances are computed using the inspection module of scikit-learn. and global shapely values are the mean of shapely values of the whole dataset. Most important and least important features are noted from these two XAI approaches. These attributions are then converted into percentages and the threshold for choosing least improtant features is **0.2 %**. The threshold is variable and can be set from the configuration file. As both techniques have very different intuition in calculating feature importances so the feature was decided unimportant only if it was suggested by both approaches.

Global Explanations do not need to be generated at the run time because they will only vary if the model or dataset changes. Therefore, global feature importances are computed and stored in the database. A script is written that will update the feature importances given different model or dataset.

6 Challenges and Limitations

A few challenges faced in the project and the limitations of approaches being used are discussed below.

The dataset analysis shows that some of the features in the model have very skewed entries. Age of some patients is in negative integers. It could be due to clerical error or dummy data. A few drugs have dosages administered in thousands of grams which is peculiar. Domain knowledge engineering is required to clean this data. Also, there are some duplicates in the data i.e. Multiple patient instances with same features are present.

KernelExplainer being used to generate shapely values is comparatively slower than TreeShap and other algorithms because they are model specific optimized algorithms but do not support the boosted random forest model that we are using.

For counterfactuals, *generic* sampling returns the most optimized counterfactual but for some patients it takes too much time because it keeps find better counterfactuals. It can take up to one minute or 10 minutes to generate counterfactual with *generic* sampling. To cater this, a different sampling method *random* is used. This sampling algorithm is, sometimes, unable to find any counterfactual in the first try. It might be because of not having enough data for a cancer type or may be it is not realistically possible to generate counterfactual for that instance. However, a loop is used to try finding CF for 5 times. If it does not find any CF, none is returned.

7 Results

7.1 Dataset Insights

Training sets of each cancer type were analyzed and some of the insights are discussed. The prevalence of failed plans in Colon, Lung, and Breast cancer is shown in Fig 1.

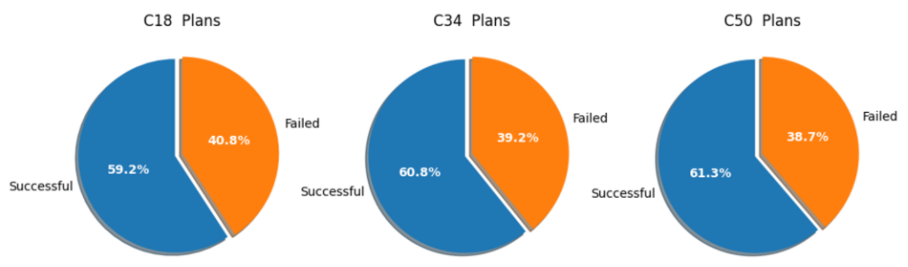


Fig. 1. Plan Failure ratio in data sets

Analyzing gender prevalence is also important to avoid any gender bias in the model. Analysis (Fig 2) shows that there is no bias in the gender feature of the model. C50 - Breast cancer has a prevalence of 97% for female patients which is also normal as this cancer type is common in women.

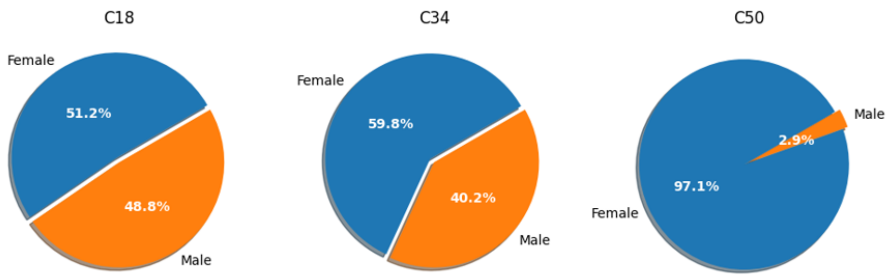


Fig. 2. Gender Prevalence in data sets

Another insight might be the distribution of the age of patients and their administered drugs dosages. Age histogram of three cancer types shows that it follows the normal distribution hence, imparts no bias in the model as shown in the Fig 3.

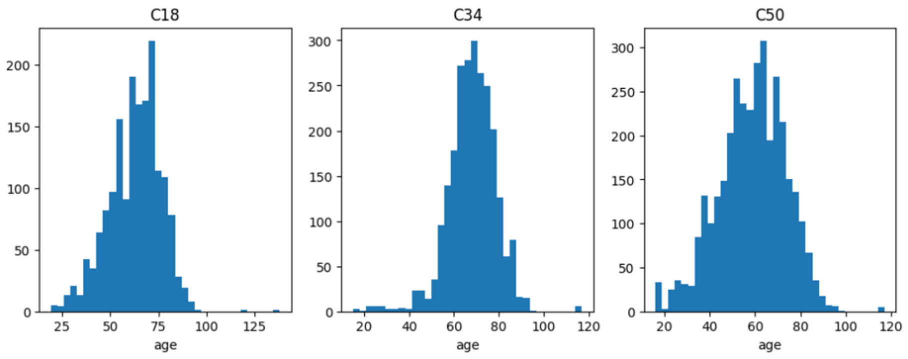


Fig. 3. Distribution of Age Feature in different cancer types

Analysis reveals that some of planned and administered drugs has dosages in thousands of grams which is another interesting insight, shown in Fig 4

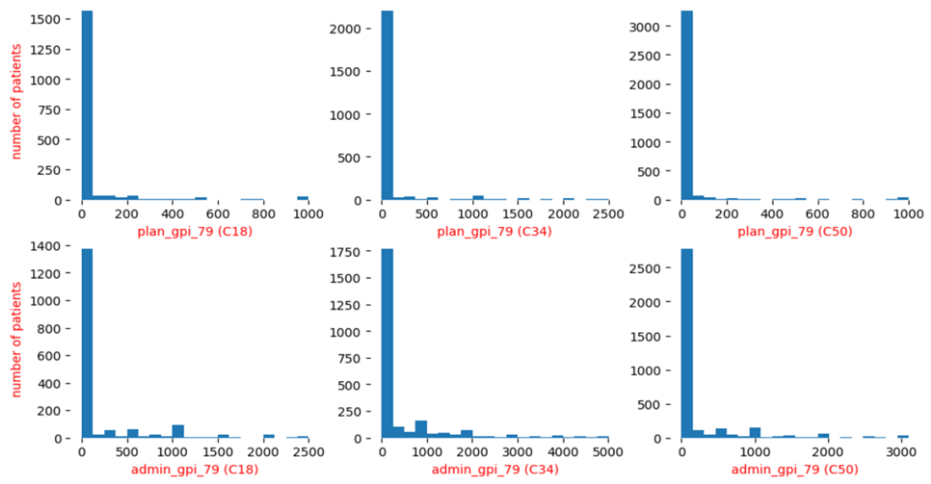


Fig. 4. Distribution of gpi-79 drug in different cancer types

7.2 Data Imputation

Data sets were imputed using multivariate imputer of scikit-learn and the models of all cancer types were retrained on the imputed data sets. Table 1 compares the new model's performance with the original models.

Table 1. Original models vs models with Imputed data sets

Cancer Type	**Original accuracy (%)**	**Imputed dataset accuracy (%)**
C18	83.33	80.46
C34	80.99	81.4
C50	86.49	83.91
C61	80.62	80.0
C90	85.71	85.0

7.3 Feedback from Feature Importance

Global feature importances were calculated for each cancer type and features having importance less than **0.2%** were dropped from each data set. New models were trained using the less-features data sets and performance metrics were recorded. Original number of features, dropped features and model performances for different cancer types are shown in Table 2.

Table 2. Models with Imputed data sets

Cancer Type	Original no. of Features	New Features	Model Performance
C18	51	43	No change
C34	61	45	-1.65%
C50	74	53	- 2.29%
C61	64	42	+0.63%
C90	70	45	No change

For C18 and C90, the model's accuracy remained the same even with simpler version of models using less features. For C61, the models accuracy was improved by **0.63%**. For C34 and C50, the accuracy decreased.

7.4 Shap

Three different explanations are being generated from the shapely values. A patient diagnosed with colon cancer is picked at random to generate shap explanations. Original TF probability is **79.42%**

Percentage Feature importance: Top ten most important features in the treat failure probability for the patient (id-8) are shown in Table 3.

Table 3. Percentage Feature attributions

Feature Name	Percentage Importance (%)
admin-gpi-22	13.79
tumor-size	11.46
stage	10.3
grade	9.39
admin-gpi-2136	8.45
admin-gpi-50	6.23
srcr	5.91
admin-gpi-41	5.81
elix-rdmsn-scr	4.85
plan-gpi-2175	4.64

Risky Features: Features that are causing % increase in the failure probability are shown in Table 4.

Table 4. Features increasing failure probability

Features	% increase in TF probability
admin-gpi-22	7.81
stage	5.83
grade	5.31
admin-gpi-2136	4.78
admin-gpi-50	3.53

Safe Features: Features that are causing % decrease in the failure probability are shown in Table 5.

Table 5. Features decreasing failure probability

Features	% decrease in TF probability
tumor-size	6.49
bsa	2.42
admin-gpi-2130	0.61
plan-gpi-2130	0.36
admin-gpi-2110	0.14

7.5 CounterFactuals

The Counterfactual module generates diverse counterfactual for the patients with original failure probability greater than 50%. Same patient as in the previous section, was analysed and different counterfactuals were generated. Counterfactual shown in Table 6. is expected to decrease the plan failure probability by **35.89%** and the new failure probability becomes **43.53%**

Table 6. Diverse Counterfactual

Features	Planned Dosage	Recommended Dosage
plan-gpi-2130	4.72	4
plan-gpi-2133	0.27	1.07
plan-gpi-2135	0	2.1
plan-gpi-2155	0.3	0
plan-gpi-2175	0.67	1.07
plan-gpi-22	0	0.3
plan-gpi-50	0.02	0

8 Application Architecture

Application architecture describes how explainability interface is designed, developed and implemented. Tentative user interface for interpretability of TF predictions was built. All the modules were written in python Language. Web framework and python libraries/dependencies used in the application are discussed below.

8.1 Flask Framework

Flask web framework is used to develop a simple front-end that allows user to generate local and global explanations for TF predictions. Flask framework has small and easy to learn API which makes it a popular web framework for simple web applications. It does not provide opinionated structure or a lot of built-in functionality, because it is designed to be easy to use and simple to understand.

8.2 Python Modules

The python packages used to develop this interface are listed below.

1. **flask** : This module includes all the tools and libraries that are used to build web application on flask framework.
2. **joblib** : joblib is a set of tools to provide lightweight pipelining in python.[2]. It reduces the redundant steps in the code and execute parallization to fully utilize all cores of CPU.
3. **pandas** : It is an open source library used for data analysis and machine learning tasks.
4. **numpy** : Numpy is a very efficient tool for scientific computing in python. Its ability to perform complex calculations with arrays and matrices makes it a powerful python tool.
5. **shap** : Shapely Additive Explanation is a module developed on the basis of game theory to explain the output of any machine learning model.

6. **dice-ml** : This module provides features-perturbed diverse counterfactuals for any test instance.
7. **sklearn** : Scikit-learn is robust and powerful python library for machine learning. It includes all types of classifier and regression models. It has built in performance metrics scorer, inspection tools, data analysis tools etc.
8. **configparser** : ConfigParser is a Python class which implements a basic configuration language for Python programs. ConfigParser allows easy customization to write python scripts.
9. **warnings** : This module shows the warnings as messages about not-so-serious errors, anomalies and bad practices in python. It does not disrupt the program flow and only shows some messages.
10. **json** : A built-in python package json can be used to work with JSON data which is in the form of key and value pairs.

8.3 Architecture Diagram

Architecture diagram of application is shown in Fig 5.

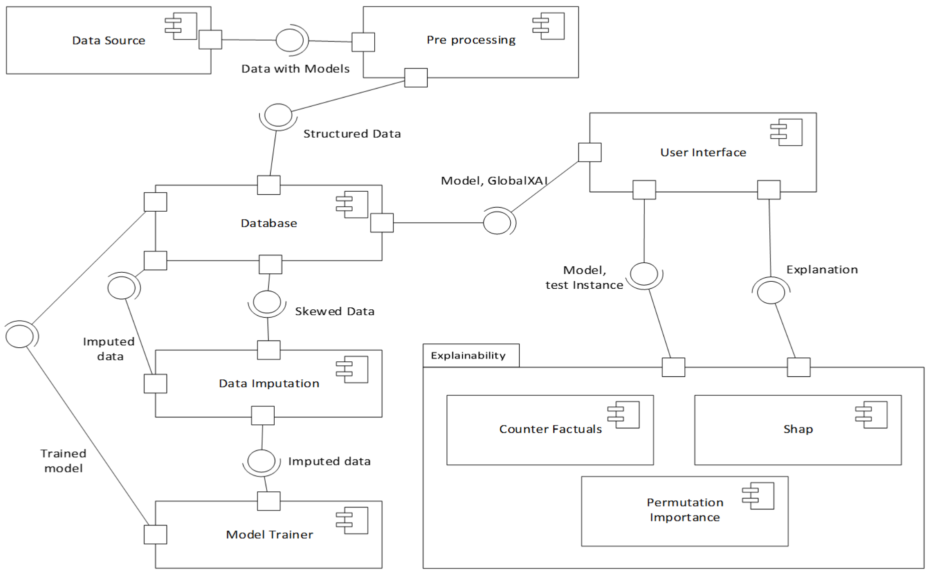


Fig. 5. Application Architecture Diagram

8.4 Class Diagram

Class diagram of the application is shown in the Fig 6

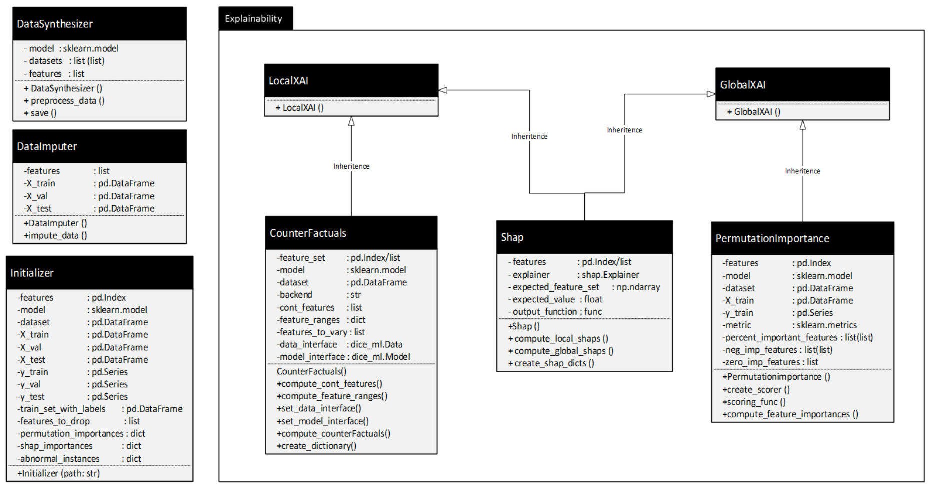


Fig. 6. Application Class Diagram

9 User Interface

A simple user interface has been developed for the explainability interface of TF model. It has three main components.

9.1 Models Comparison

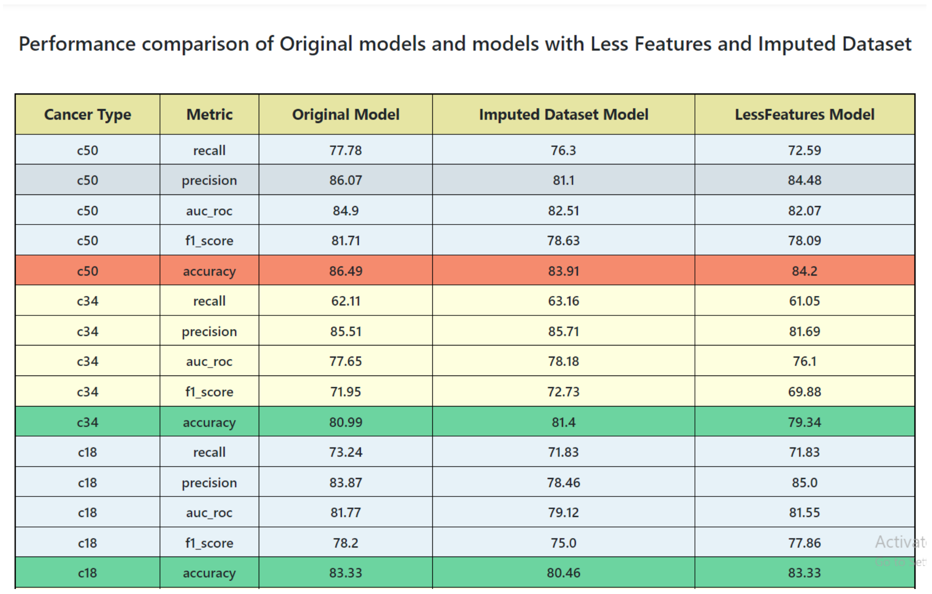
This component provides a performance comparison between original models, new models with imputed data sets and new models using less features; for each cancer type. Tentative UI is shown in the Fig. 7

Accuracy of different cancer types is highlighted in green if the imputation or using less-features have resulted in better models for them.

9.2 GlobalXAI

Global explanations for each cancer type are shown in this section of the application. Global feature importances computed by permutation feature importance and shap are shown in pi charts (Fig 8 and Fig 9) and top risky features that are driving failure risk are displayed in a bar chart. (Fig 10) General dataset insights are also shown in the Figure 11.

All of these insights are directly loaded from the database.



Performance comparison of Original models and models with Less Features and Imputed Dataset

Cancer Type	Metric	Original Model	Imputed Dataset Model	LessFeatures Model
c50	recall	77.78	76.3	72.59
c50	precision	86.07	81.1	84.48
c50	auc_roc	84.9	82.51	82.07
c50	f1_score	81.71	78.63	78.09
c50	accuracy	86.49	83.91	84.2
c34	recall	62.11	63.16	61.05
c34	precision	85.51	85.71	81.69
c34	auc_roc	77.65	78.18	76.1
c34	f1_score	71.95	72.73	69.88
c34	accuracy	80.99	81.4	79.34
c18	recall	73.24	71.83	71.83
c18	precision	83.87	78.46	85.0
c18	auc_roc	81.77	79.12	81.55
c18	f1_score	78.2	75.0	77.86
c18	accuracy	83.33	80.46	83.33

Fig. 7. Performance Comparison of different Models

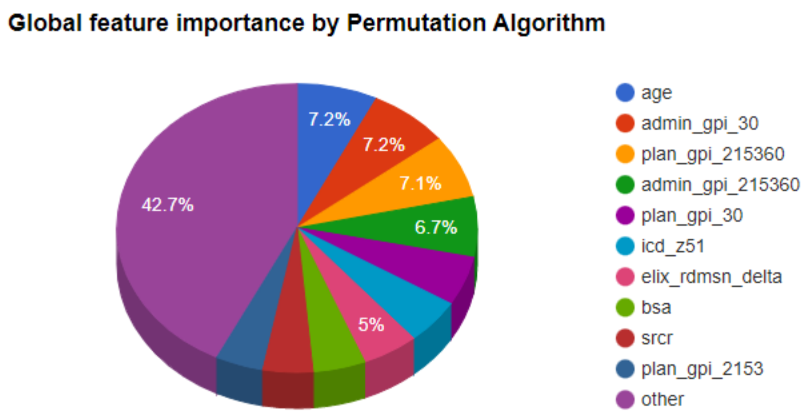


Fig. 8. Global permutation feature attributions (C18 - colon)

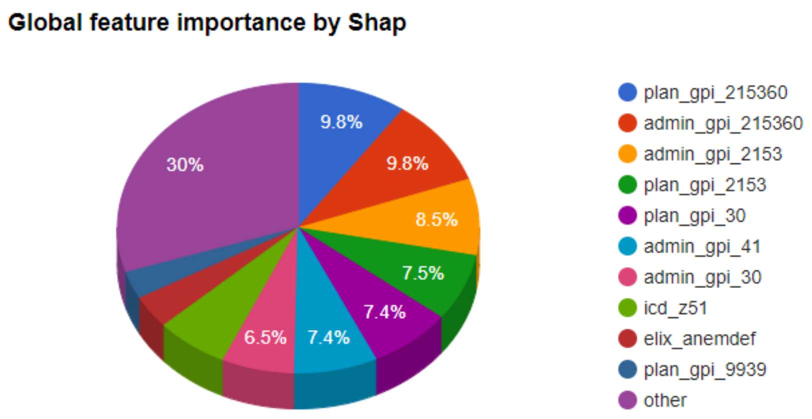


Fig. 9. Global shap feature attributions (C18 - colon)

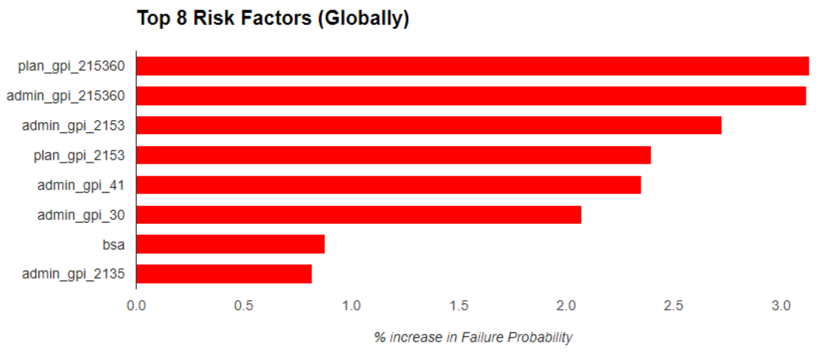
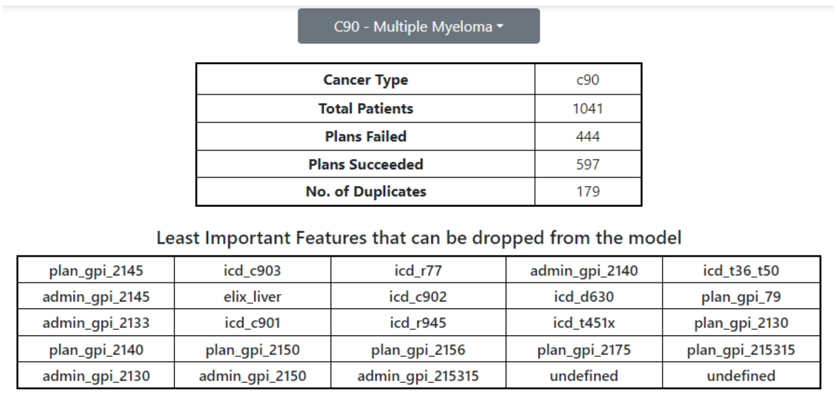


Fig. 10. Top 8 features driving failure risk (C18 - colon)



Cancer Type	c90
Total Patients	1041
Plans Failed	444
Plans Succeeded	597
No. of Duplicates	179

Least Important Features that can be dropped from the model

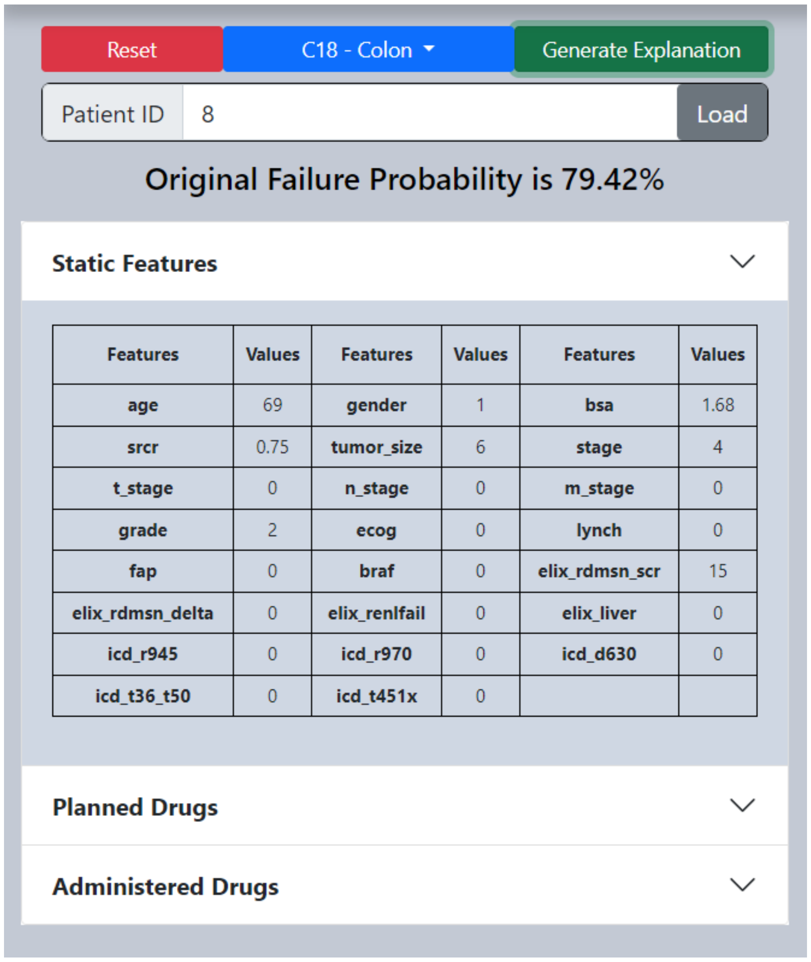
plan_gpi_2145	icd_c903	icd_r77	admin_gpi_2140	icd_t36_t50
admin_gpi_2145	elix_liver	icd_c902	icd_d630	plan_gpi_79
admin_gpi_2133	icd_c901	icd_r945	icd_t451x	plan_gpi_2130
plan_gpi_2140	plan_gpi_2150	plan_gpi_2156	plan_gpi_2175	plan_gpi_215315
admin_gpi_2130	admin_gpi_2150	admin_gpi_215315	undefined	undefined

Fig. 11. General data insights for Colon Cancer

9.3 LocalXAI

Local explanation interface consists of dropdown to select cancer type, and an input field to enter patient id. Both the model and patient will be loaded on the UI and displayed as shown in Fig. 12. 'Generate Explanation' Button will generate local shap and counterfactual explanations for that patient. Counterfactuals will not be computed if the original failure probability of that particular patient is less than 50%. Shap risky and safe features are shown in Fig. 14 and general importance of features in the failure prediction is shown in a pi chart in Fig. 13

CounterFactuals are shown separately in the table below the shap charts. (Fig. 15)



Original Failure Probability is 79.42%

Static Features

Features	Values	Features	Values	Features	Values
age	69	gender	1	bsa	1.68
srcr	0.75	tumor_size	6	stage	4
t_stage	0	n_stage	0	m_stage	0
grade	2	ecog	0	lynch	0
fap	0	braf	0	elix_rdmsn_scr	15
elix_rdmsn_delta	0	elix_renlfail	0	elix_liver	0
icd_r945	0	icd_r970	0	icd_d630	0
icd_t36_t50	0	icd_t451x	0		

Planned Drugs

Administered Drugs

Fig. 12. Patient Features Interface

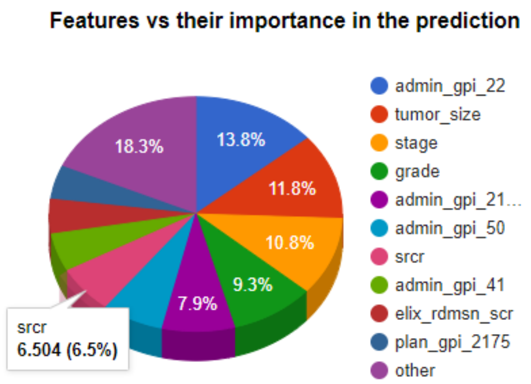


Fig. 13. Local Feature importance by Shap

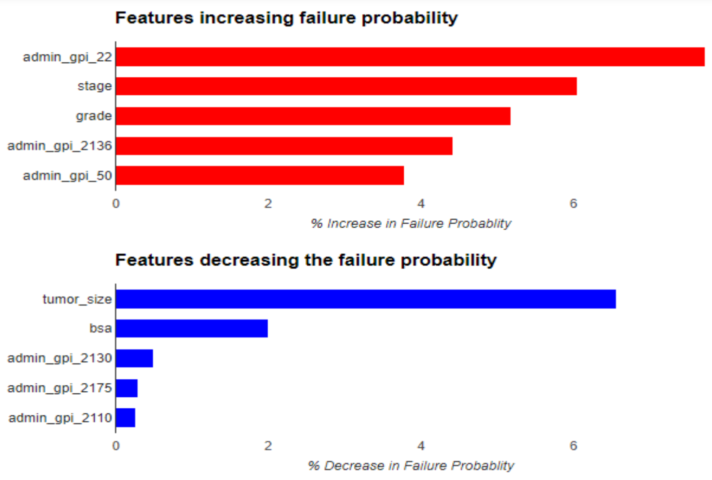
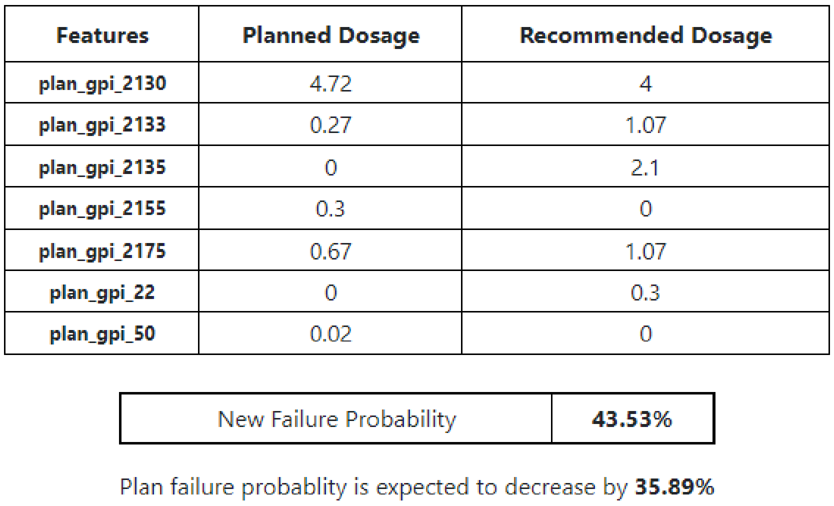


Fig. 14. Risky and Safe features by Shap



Features	Planned Dosage	Recommended Dosage
plan_gpi_2130	4.72	4
plan_gpi_2133	0.27	1.07
plan_gpi_2135	0	2.1
plan_gpi_2155	0.3	0
plan_gpi_2175	0.67	1.07
plan_gpi_22	0	0.3
plan_gpi_50	0.02	0

New Failure Probability	**43.53%**

Plan failure probablity is expected to decrease by **35.89%**

Fig. 15. CounterFactual explanation

10 Conclusion

Model agnostic approaches of Explainable AI were successfully implemented to create an interpretability interface for a machine learning model that predicts cancer treatment failure probability. The project involved analyzing the dataset, cleaning and imputing the data, implementing Shap and counterfactuals explanations for local explanations, generating global explanations using permutation feature importance and Shap global, and developing a front-end app with features for models comparison, global explanation interface, and local explanation interface. The interpretability interface developed in this project improves trust and understanding of machine learning models for predicting cancer treatment failure probability.

11 Future Work

There are several recommendations and future work that can be considered for this project.

1. The project can be extended to include other types of cancer and different prediction tasks such as survival analysis. This can provide more insights into cancer treatment and help develop better treatment strategies.
2. The explainability interface can be further improved by incorporating other types of explanations i.e. What-else. and different types of plots i.e. partial dependence plots, decision plots. This can provide additional insights into the model's behavior and make the interface more comprehensive.

3. The data set analysis uncovered few abnormal distribution in some features and duplication issue in the data. This feedback can be used in the data cleaning and retraining the models to improve the performance of models and XAI interface.
4. The front-end app can be improved by incorporating user feedback and making it more interactive and user-friendly. This can increase the usability and adoption of the interface by healthcare professionals and patients.
5. Counterfactual strategy can be improved by in-depth domain analysis of different features and their permissible range.

12 References

References

1. Biswal, A.: https://www.simplilearn.com/tutorials/artificial-intelligence-tutorial/artificial-intelligence-applications
2. Joblib: running python functions as pipeline job, https://joblib.readthedocs.io/en/latest/
3. Molnar, C.: Interpretable Machine Learning. Christoph Molnar (2019)
4. Permutation feature importance, https://scikit-learn.org/stable/modules/permutation_importance.html
5. Scott Lundberg, S.I.L.: Shap (2018), https://github.com/slundberg/shap
6. Sharma, A.: Diverse counterfactual explanations (dice) for ml (2020), https://github.com/interpretml/DiCE
7. Sina Mohseni, Niloofar Zarei, E.D.R.: A multidisciplinary survey and framework for design and evaluation of explainable ai systems (2021)